Predicting NBA Game Outcomes Based on Basketball Statistics of An Individual Player

Neel Pendyala

Final Project STP 598

Arizona State University

May 5, 2018

Introduction & Background

Due to the advent of new technology and innovation, the National Basketball League (NBA) has been able to collect information about players' basketball performance and ability that was not previously accessible. Hundreds of new metrics and measures have been developed as a result of state-of-the-art tracking technology which is able to capture even the most specific details of player performance, such as the total distance traveled by a player while playing in a game. The abundance of new data, however, has made it difficult to identify the information that best serves the purpose of NBA basketball teams, which is to win as many games as possible in order to compete for the NBA championship. One of the tasks of NBA teams is determining which areas of a player's performance are most beneficial and detrimental to the team's success. While this can be accomplished through other methods such as video analysis, this can be a time-consuming process. A more efficient alternative could be to identify which player basketball statistics most strongly influence the outcome of games (win or loss). With this method, coaches would be able to determine in which areas players need to improve so that the team has a greater chance at success. The objective of this project was to identify player basketball statistics that have the greatest impact on the outcome of NBA basketball games. Forty-eight metrics of an individual player's performance were collected for 10,0140 NBA games. Whether the game resulted in a win or loss for that player was also observed. These forty-eight basketball statistics for a single player were used to predict whether NBA basketball games resulted in a win or loss using a regularized logistic regression model.

Methods

Data was collected on 48 basketball statistics for 10,0140 instances. Each instance represents the player's performance during an NBA playoff game. The outcome of the game – whether it resulted in a win or loss – was also recorded. There were 5,122 instances which resulted in a win, and 5,018 instances which resulted in a loss. Therefore, class imbalance was not an issue. In all instances, the player played for at least 20 minutes during the game. Players who played less than 20 minutes were excluded as they are less likely to have had an impact on the outcome of the game. NBA Stats and Basketball-Reference were the online sources used for data collection. Initial inspection of the data indicated that several of the basketball statistics were similar metrics of

performance, such as `FGA` and `%FGA`, which represent field goals attempted, and percent of team's field goals attempted while on the court, respectively. These similar basketball statistics were likely to be correlated; this was confirmed by the correlation plot shown in Appendix 1. To address the multicollinearity within the predictor variables, principal components analysis (PCA) was employed to not only address the multicollinearity, but to also reduce the dimensionality of the data. The separation in the data achieved by PCA is shown in Fig. 1.

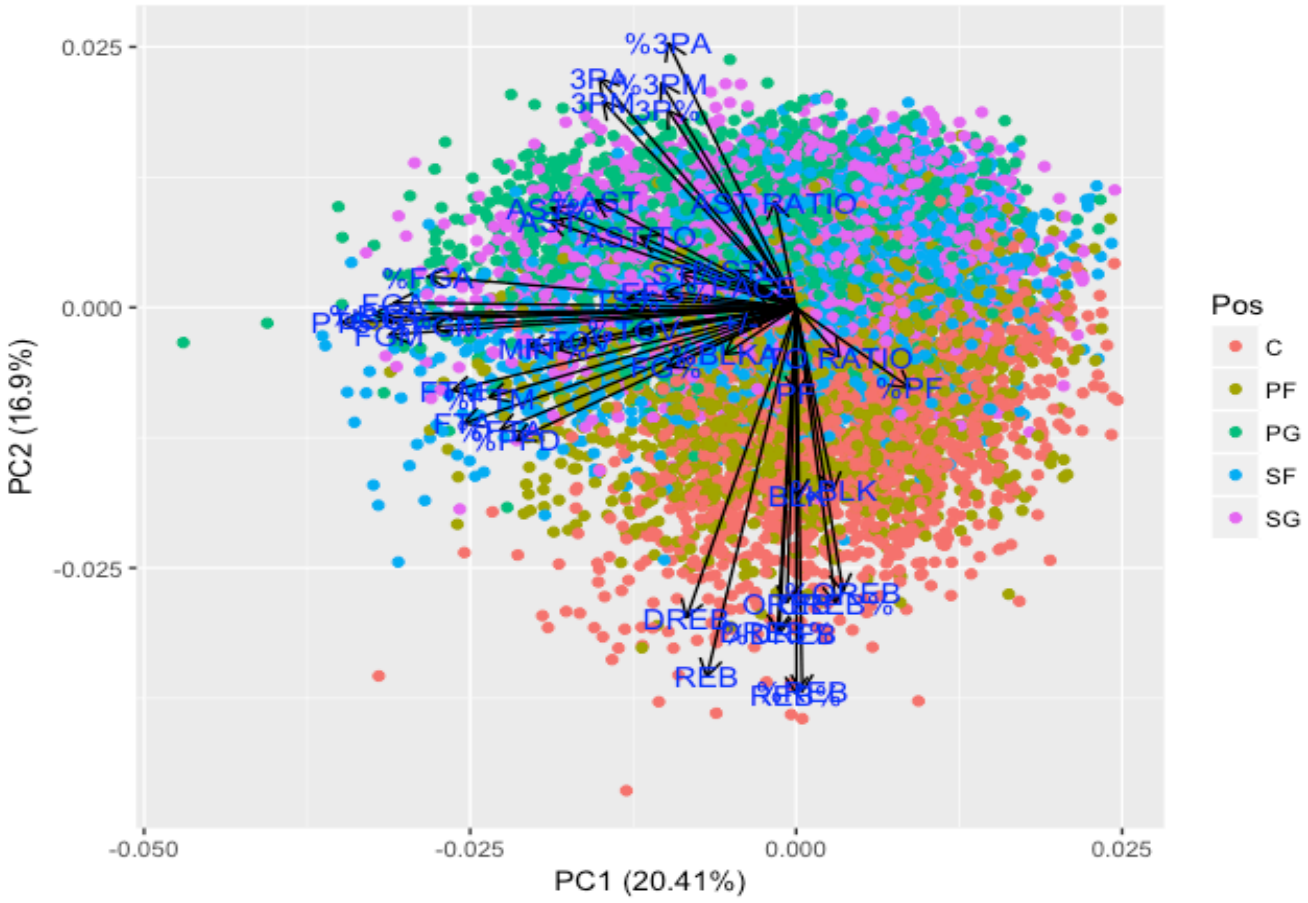


Figure 1. Two principal components projected onto data coded by player position

The plot shows that individual basketball statistics are dependent on player position. There are five possible positions: point-guard (n=2,032), shooting guard (n=2,402), small forward (n=2,058), power forward (n=1,902), and center (n=1,746). Due to this association between player position and player statistics, the position of the player was also used to predict NBA

game outcomes along with his individual basketball statistics. Principal components attempt to capture the variance of the data. A greater amount of variance is explained with each additional principal component (Fig. 2).

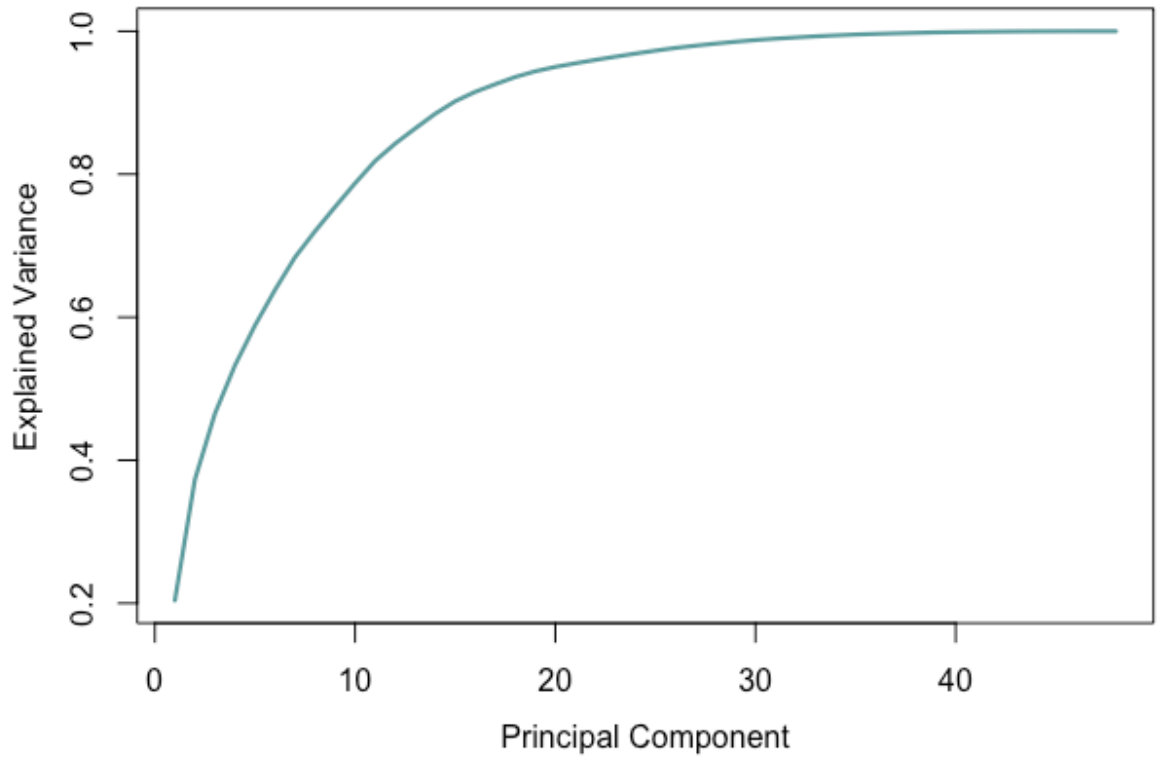


Figure 2. Cumulative explained variance given number of components

With PCA there is the risk of losing vital predictive information in an attempt to decorrelate the data. It is therefore important to pick the number of principal components that leads to the best out-of-sample performance for the given model. The ability to predict the outcome of NBA games given the player's position and individual basketball statistics was tested using a logistic regression model with L1 penalty. The ability of players' individual basketball statistics in the form of principal components and player position data to predict the outcomes of NBA games was tested using a logistic regression model with L1 penalty. This was tested for 10 to 45 principal components (out of 48 total principal components). For each set of principal components the model was run using 20 different randomly sampled train and test splits given 10 different values for the L1 penalization parameter, lambda. The out-of-sample area under the

curve (AUC) was obtained for each value of lambda, and each set of principal components. The number of principal components and lambda value which gave the greatest out-of-sample AUC were used to build the final model, which was then tested on a new, validation set. The performance of the model on the validation set was evaluated.

Results

The regularized logistic regression model predicted NBA game outcomes using 10 different values for lambda and given a number of principal components within the range of 10 to 45. The AUC was calculated and recorded for each principal component and lambda value pairing. All out-of-sample AUC scores and lambda parameter values are listed in Appendix 2 and 3. The out-of-sample AUC was averaged across all 10 lambda values to obtain the average out-of-sample AUC for each set of principal components between 10 and 45. Fig. 3 illustrates the incremental improvement in training and testing AUC with increasing number of principal components.

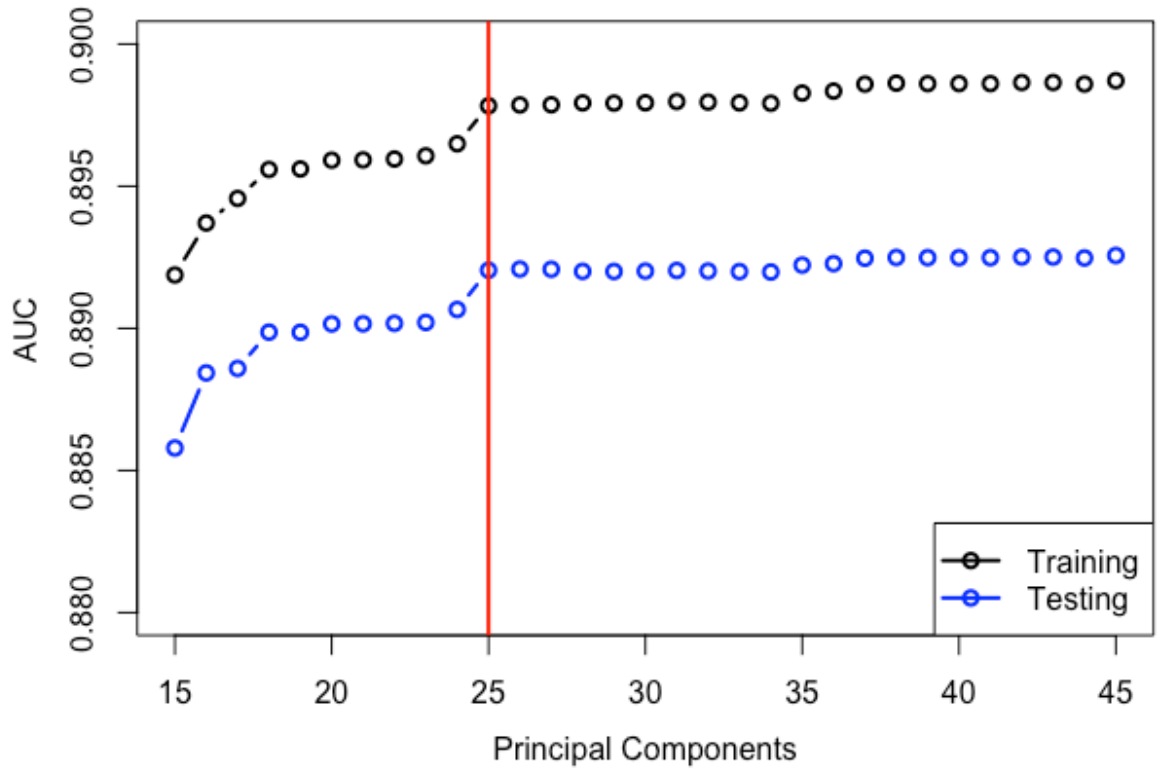


Figure 3. Training and testing AUC given number of principal components between 15 and 45

As indicated by the plot, there is a limit to the model's predictive ability as the AUC fails to increase by any meaningful amount after 25 principal components. Given a larger number of features, models tend to become more complex, and less generalizable. Therefore, 25 was chosen as the number of principal components to include in the final model. This gave an out-of-sample AUC equal to 0.897. The coefficients for the finalized model fit are shown in Appendix 4.
To confirm that the model was not overfitting, it was used to predict on an unseen validation dataset (n=603). The model performed reasonably well as indicated by the ROC Curve (Fig. 4) and the confusion matrix (Fig. 5).

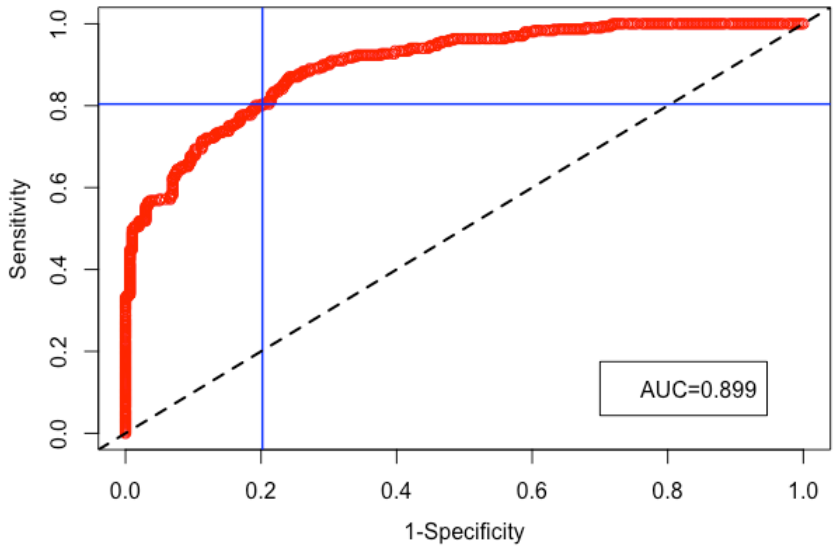


Figure 4. ROC Curve for model prediction on validation set

Figure 5. Normalized confusion matrix of observed versus predicted game outcomes using validation set

Model prediction on the validation set gave an AUC of 0.899. The blue lines in Fig. 4 represent the true positive rate (TPR=0.804) and false positive rate (FPR=0.202) given a decision threshold equal to 0.5. While using a different decision threshold may lead to slightly better classification accuracy, a threshold of 0.5 is the most intuitive since this can be viewed as the percent chance for a particular outcome. Greater than 50% chance translates to a win; lower than 50% corresponds to a loss. Fig. 5 shows the normalized confusion matrix when this decision threshold of 0.5 is used. Classification accuracy comes out to approximately 80%, so out of 603 instances, 120 were not correctly classified.

Discussion

The original objective was to identify which individual player basketball statistics are most important to winning. The data was scaled prior to the analysis, and therefore the coefficients of the model can be treated as measures of relative importance with respect to the outcome of the game. The general model can formulaically be represented as shown below,

$$\hat{p} = \beta_0 + \beta_1 PF + \beta_2 PG + \beta_3 SF + \beta_4 SG + \beta_5 \begin{pmatrix} \phi_{1,1} x_1 \\ + \\ \phi_{2,1} x_2 \\ + \\ \vdots \\ + \\ \phi_{48,1} x_{48} \end{pmatrix} + \cdots + \beta_{52} \begin{pmatrix} \phi_{1,48} x_1 \\ + \\ \phi_{2,48} x_2 \\ + \\ \vdots \\ + \\ \phi_{48,48} x_{48} \end{pmatrix}$$

where β_{0-52} equal the intercept and coefficient constants given by the logistic regression model; PF, PG, SF, SG represent the player's position and can only take on the values of zeros or ones; $\phi_{1,1-48,48}$ equal the coefficients generated by PCA; x_{1-48} represent the 48 basketball statistics. For the final model, not all principal components were used and few of the variables were given a coefficient of zero. This simplified model along with the coefficient values is shown below.

$$\hat{p} = 0.053 - \ 0.063PG + 0.094SF - 0.073\begin{pmatrix} -0.178x_1 \\ + \\ -0.301x_2 \\ + \\ \vdots \\ + \\ -0.28x_{48} \end{pmatrix} + \cdots + 0.417\begin{pmatrix} \phi_{1,25}x_1 \\ + \\ \phi_{2,25}x_2 \\ + \\ \vdots \\ + \\ \phi_{48,25}x_{48} \end{pmatrix}$$

Distributing the model coefficients can further simplify the formula so that each variable has a single coefficient. The absolute values of the coefficients for the 48 basketball statistics serve as the relative importance measures and are shown in Fig. 6.

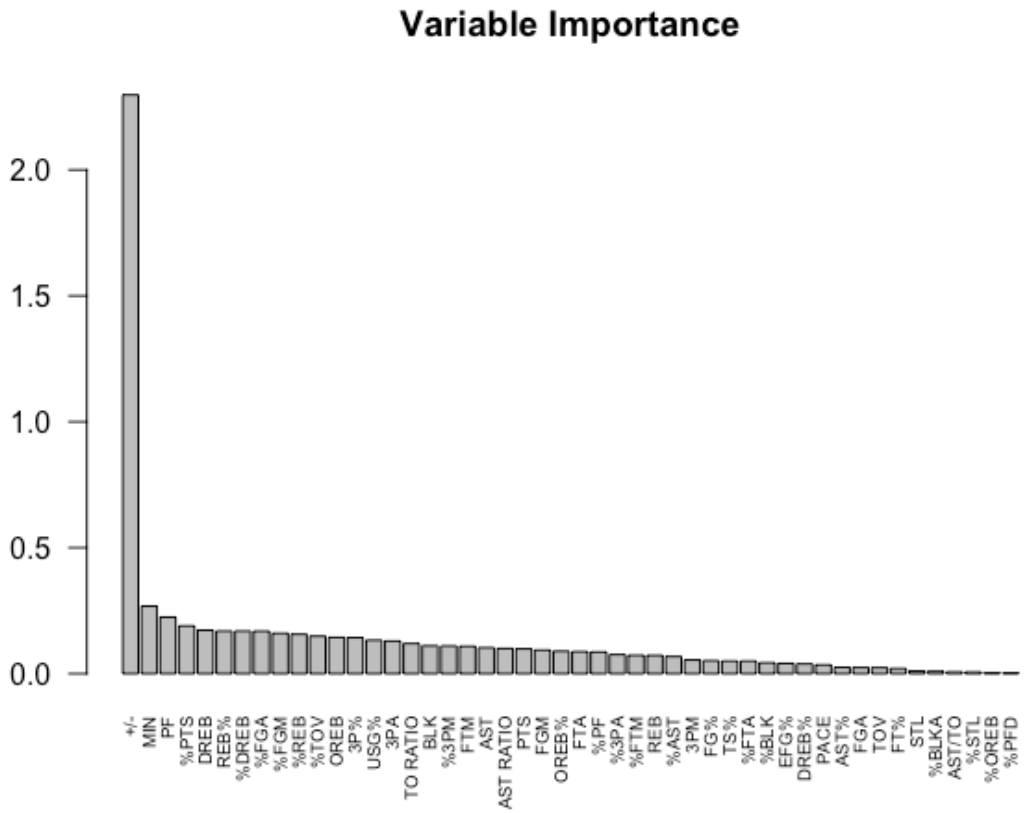
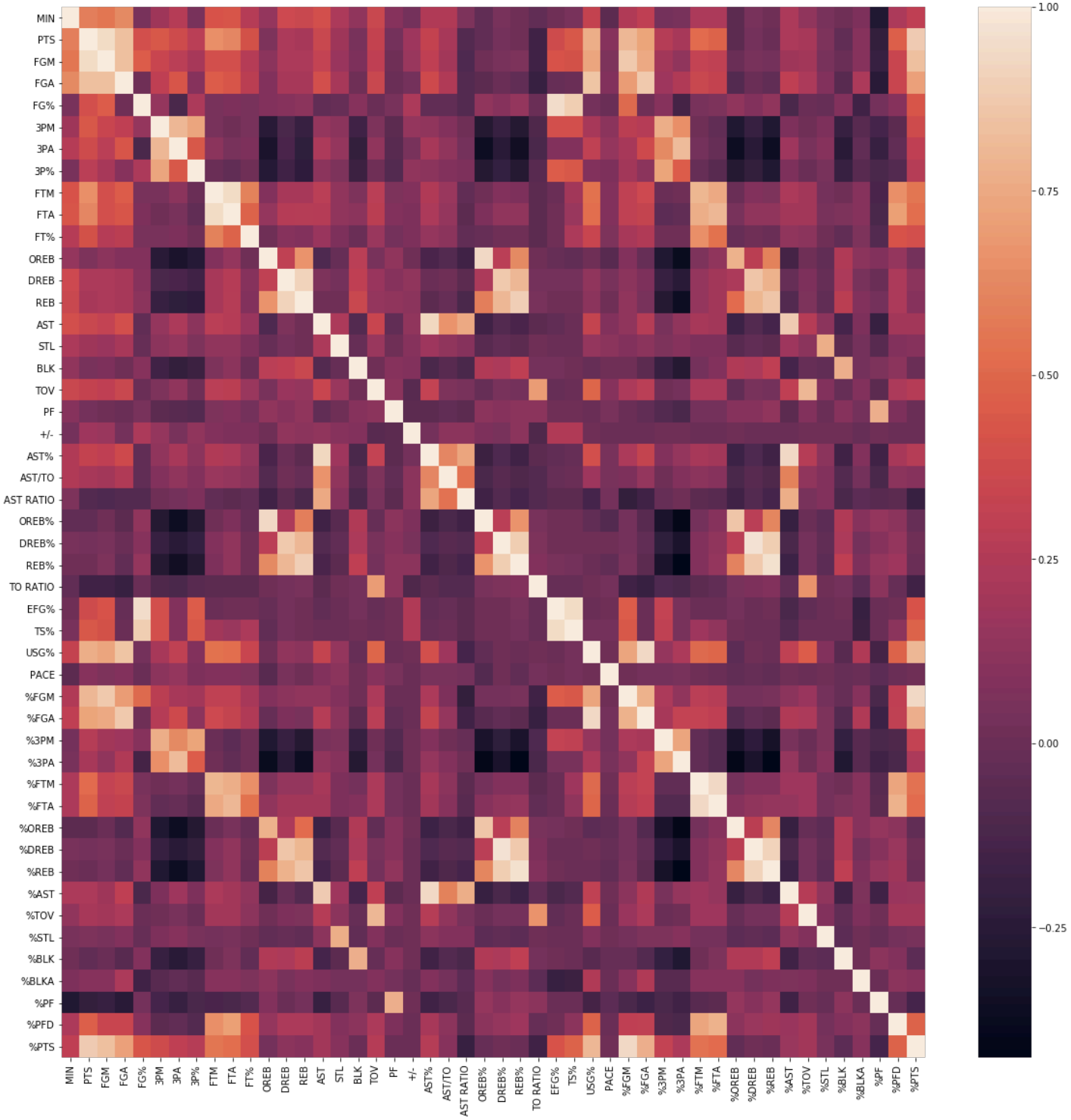


Figure 6. Relative variable importance for the 48 basketball statistics included in model

The figure shows one variable as being significantly more important than the others. This variable is the basketball metric known as Plus-Minus rating (+/-). Plus-Minus rating measures

the number of points by which one team has outscored the other while a particular player is on the court. Plus-Minus rating is a widely referenced statistic in basketball and is commonly used to evaluate the importance and impact of a player to his team. This model further verifies the robustness of Plus-Minus rating by demonstrating that it is a strong determinant of NBA game outcomes. However, besides this one measure, no other basketball statistic provided any significant importance. Even the variables that were chosen as being important were not consistent with intuition. For instance, according to the model, the number of minutes played during the game (MIN) has the second largest role in whether the game results in a win or loss. The model indicates that playing more minutes decreases the team's chance of winning. This is more likely a pattern in the data, as opposed to a meaningful association between the two variables. Considering all variables except for Plus-Minus rating did so poorly in terms of predicting wins or losses, variables with greater amount of association with game outcomes should be used in the future. The model should also test the effect of interaction terms between variables that are dependent on each other such as MIN and PTS on predictive performance. Interaction terms could also be created between the players' position variable and basketball statistics that tend to vary based on the players' position, such as number of rebounds (REB). The model was able to justify the use of Plus-Minus, the gold standard in today's basketball game, as a measure for evaluating player impact. In the future, it will be of interest to explore the other possible basketball metrics of an individual player that best determine whether an NBA basketball game will be won or lost.

Appendix



Appendix 1. Heatmap of correlation matrix for 48 basketball statistics

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
10	0.6719168	0.6713627	0.6708019	0.6703786	0.6701028	0.6696758	0.6692537	0.6688435	0.6682808	0.6676969
11	0.6757675	0.6750709	0.6744721	0.6740880	0.6738091	0.6734406	0.6730615	0.6726382	0.6720960	0.6714754
12	0.6837217	0.6834136	0.6830307	0.6828589	0.6825944	0.6823485	0.6820503	0.6817677	0.6814702	0.6810662
13	0.6849530	0.6846884	0.6844108	0.6840249	0.6837815	0.6834740	0.6831434	0.6828091	0.6824786	0.6820223
14	0.8591512	0.8590124	0.8589644	0.8589128	0.8588630	0.8587186	0.8585648	0.8584528	0.8582704	0.8580612
15	0.8862477	0.8861456	0.8860192	0.8859482	0.8858636	0.8857584	0.8856787	0.8855523	0.8854552	0.8852224
16	0.8891833	0.8889748	0.8888515	0.8887675	0.8886206	0.8884170	0.8882135	0.8880056	0.8877728	0.8875026
17	0.8890601	0.8889592	0.8888609	0.8888030	0.8887812	0.8886212	0.8884911	0.8883037	0.8881014	0.8878923
18	0.8904538	0.8903293	0.8902167	0.8901519	0.8900492	0.8898886	0.8896726	0.8895145	0.8893134	0.8890532
19	0.8903872	0.8903256	0.8902104	0.8901582	0.8900486	0.8898961	0.8896676	0.8895213	0.8893141	0.8890532
20	0.8909008	0.8907402	0.8905678	0.8904395	0.8903318	0.8901606	0.8899434	0.8896851	0.8894585	0.8893091
21	0.8909244	0.8907290	0.8905584	0.8904526	0.8903374	0.8901625	0.8899509	0.8896869	0.8894641	0.8893091
22	0.8908908	0.8907676	0.8906063	0.8905223	0.8903555	0.8901712	0.8899596	0.8896975	0.8894659	0.8893091
23	0.8909145	0.8908006	0.8906798	0.8905833	0.8904140	0.8902484	0.8899490	0.8896975	0.8894659	0.8893091
24	0.8916721	0.8915301	0.8913004	0.8911056	0.8908815	0.8906649	0.8903598	0.8900031	0.8896633	0.8894298
25	0.8930565	0.8929681	0.8927876	0.8925940	0.8923630	0.8920648	0.8917480	0.8913795	0.8909854	0.8904912
26	0.8932532	0.8930901	0.8928685	0.8925884	0.8923630	0.8920648	0.8917480	0.8913795	0.8909854	0.8904912
27	0.8932594	0.8931044	0.8928442	0.8925659	0.8923493	0.8920586	0.8917368	0.8913701	0.8909798	0.8904912
28	0.8929656	0.8928934	0.8927066	0.8924956	0.8922896	0.8920487	0.8917623	0.8913882	0.8909854	0.8904912
29	0.8929650	0.8928791	0.8927048	0.8924925	0.8922840	0.8920555	0.8917530	0.8913851	0.8909798	0.8904912
30	0.8930832	0.8928921	0.8927048	0.8924925	0.8922840	0.8920555	0.8917530	0.8913851	0.8909798	0.8904912
31	0.8931081	0.8929911	0.8927365	0.8925379	0.8922896	0.8920487	0.8917623	0.8913882	0.8909854	0.8904912
32	0.8930720	0.8929394	0.8927017	0.8925068	0.8922815	0.8920505	0.8917430	0.8913751	0.8909805	0.8904912
33	0.8930397	0.8928834	0.8926886	0.8924794	0.8922678	0.8920375	0.8917243	0.8913714	0.8909686	0.8904912
34	0.8930104	0.8928542	0.8926687	0.8924688	0.8922572	0.8920157	0.8916982	0.8913695	0.8909637	0.8904912
35	0.8930366	0.8929967	0.8929482	0.8927035	0.8924838	0.8922279	0.8919852	0.8916260	0.8913023	0.8909201
36	0.8933546	0.8931237	0.8929463	0.8927197	0.8924925	0.8922454	0.8919951	0.8916372	0.8913104	0.8909201
37	0.8935109	0.8933285	0.8931243	0.8928691	0.8925921	0.8923692	0.8921445	0.8918675	0.8915781	0.8912674
38	0.8936472	0.8934181	0.8931187	0.8928710	0.8926052	0.8923811	0.8921539	0.8918793	0.8915774	0.8912674
39	0.8936478	0.8933895	0.8931243	0.8928691	0.8925921	0.8923692	0.8921445	0.8918675	0.8915781	0.8912674
40	0.8936472	0.8933895	0.8931243	0.8928691	0.8925921	0.8923692	0.8921445	0.8918675	0.8915781	0.8912674
41	0.8936472	0.8933895	0.8931243	0.8928691	0.8925921	0.8923692	0.8921445	0.8918675	0.8915781	0.8912674
42	0.8936952	0.8934318	0.8931287	0.8928984	0.8926157	0.8923966	0.8921694	0.8918800	0.8915799	0.8912674
43	0.8936952	0.8934318	0.8931287	0.8928984	0.8926157	0.8923966	0.8921694	0.8918800	0.8915799	0.8912674
44	0.8936273	0.8933764	0.8931125	0.8928517	0.8925884	0.8923587	0.8921327	0.8918569	0.8915725	0.8912674
45	0.8940487	0.8935034	0.8931386	0.8929519	0.8926220	0.8924222	0.8921812	0.8918955	0.8915893	0.8912674

Appendix 2. Out-of-sample AUC scores for 10 to 45 principal components and 10 values of lambda

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
10	0.0019045873	0.0038589711	0.005813355	0.007767739	0.009722123	0.011676506	0.013630890	0.015585274	0.017539658	0.019494042
11	0.0006844732	0.0025820033	0.004479534	0.006377064	0.008274594	0.010172124	0.012069654	0.013967184	0.015864715	0.017762245
12	0.0010898749	0.0026072807	0.004124687	0.005642092	0.007159498	0.008676904	0.010194310	0.011711716	0.013229122	0.014746528
13	0.0010898749	0.0024617206	0.003833566	0.005205412	0.006577258	0.007949104	0.009320949	0.010692795	0.012064641	0.013436487
14	0.0004533483	0.0017101428	0.002966937	0.004223732	0.005480526	0.006737321	0.007994115	0.009250910	0.010507704	0.011764499
15	0.0010472934	0.0021219690	0.003196644	0.004271320	0.005345996	0.006420671	0.007495347	0.008570022	0.009644698	0.010719373
16	0.0010472934	0.0020161601	0.002985027	0.003953893	0.004922760	0.005891627	0.006860494	0.007829360	0.008798227	0.009767094
17	0.0008694814	0.0017616959	0.002653910	0.003546125	0.004438339	0.005330554	0.006222768	0.007114983	0.008007197	0.008899412
18	0.0007218587	0.0015426314	0.002363404	0.003184177	0.004004949	0.004825722	0.005646495	0.006467267	0.007288040	0.008108813
19	0.0006577308	0.0014856288	0.002313527	0.003141425	0.003969323	0.004797221	0.005625119	0.006453017	0.007280915	0.008108813
20	0.0006577308	0.0014055883	0.002153446	0.002901303	0.003649161	0.004397018	0.005144876	0.005892733	0.006640591	0.007388448
21	0.0005992998	0.0013536496	0.002107999	0.002862349	0.003616699	0.004371049	0.005125399	0.005879748	0.006634098	0.007388448
22	0.0005460597	0.0013063250	0.002066590	0.002826856	0.003587121	0.004347387	0.005107652	0.005867917	0.006628183	0.007388448
23	0.0005460597	0.0013063250	0.002066590	0.002826856	0.003587121	0.004347387	0.005107652	0.005867917	0.006628183	0.007388448
24	0.0004975492	0.0011902747	0.001883000	0.002575726	0.003268451	0.003961177	0.004653902	0.005346628	0.006039353	0.006732079
25	0.0004533483	0.0010845340	0.001715720	0.002346905	0.002978091	0.003609277	0.004240462	0.004871648	0.005502834	0.006134019
26	0.0004533483	0.0010845340	0.001715720	0.002346905	0.002978091	0.003609277	0.004240462	0.004871648	0.005502834	0.006134019
27	0.0004975492	0.0011238237	0.001750098	0.002376373	0.003002647	0.003628922	0.004255196	0.004881470	0.005507745	0.006134019
28	0.0004533483	0.0010845340	0.001715720	0.002346905	0.002978091	0.003609277	0.004240462	0.004871648	0.005502834	0.006134019
29	0.0004975492	0.0011238237	0.001750098	0.002376373	0.003002647	0.003628922	0.004255196	0.004881470	0.005507745	0.006134019
30	0.0004975492	0.0011238237	0.001750098	0.002376373	0.003002647	0.003628922	0.004255196	0.004881470	0.005507745	0.006134019
31	0.0004533483	0.0010845340	0.001715720	0.002346905	0.002978091	0.003609277	0.004240462	0.004871648	0.005502834	0.006134019
32	0.0005460597	0.0011669441	0.001787828	0.002408713	0.003029597	0.003650482	0.004271366	0.004892251	0.005513135	0.006134019
33	0.0006577308	0.0012662073	0.001874684	0.002483160	0.003091637	0.003700113	0.004308590	0.004917066	0.005525543	0.006134019
34	0.0007218587	0.0013232099	0.001924561	0.002525912	0.003127263	0.003728615	0.004329966	0.004931317	0.005532668	0.006134019
35	0.0005992998	0.0011537209	0.001708142	0.002262563	0.002816984	0.003371405	0.003925827	0.004480248	0.005034669	0.005589090
36	0.0005460597	0.0011063964	0.001666733	0.002227070	0.002787406	0.003347743	0.003908080	0.004468417	0.005028753	0.005589090
37	0.0004975492	0.0010081072	0.001518665	0.002029223	0.002539781	0.003050339	0.003560897	0.004071455	0.004582013	0.005092571
38	0.0004533483	0.0009688175	0.001484287	0.001999756	0.002515225	0.003030694	0.003546163	0.004061632	0.004577102	0.005092571
39	0.0004975492	0.0010081072	0.001518665	0.002029223	0.002539781	0.003050339	0.003560897	0.004071455	0.004582013	0.005092571
40	0.0004975492	0.0010081072	0.001518665	0.002029223	0.002539781	0.003050339	0.003560897	0.004071455	0.004582013	0.005092571
41	0.0004975492	0.0010081072	0.001518665	0.002029223	0.002539781	0.003050339	0.003560897	0.004071455	0.004582013	0.005092571
42	0.0004130741	0.0009330182	0.001452962	0.001972906	0.002492850	0.003012794	0.003532738	0.004052683	0.004572627	0.005092571
43	0.0004130741	0.0009330182	0.001452962	0.001972906	0.002492850	0.003012794	0.003532738	0.004052683	0.004572627	0.005092571
44	0.0005460597	0.0010512276	0.001556395	0.002061563	0.002566731	0.003071899	0.003577067	0.004082235	0.004587403	0.005092571
45	0.0003124754	0.0008435971	0.001374719	0.001905841	0.002436962	0.002968084	0.003499206	0.004030327	0.004561449	0.005092571

Appendix 3. Lambda values used to test model for each set of principal components between 10 and 45

	x
(Intercept)	0.0527501
PosPF	0.0000000
PosPG	-0.0630637
PosSF	0.0938461
PosSG	0.0000000
PC1	-0.0728433
PC2	-0.0325304
PC3	-0.1687795
PC4	-0.3583973
PC5	-0.3986148
PC6	-0.1781189
PC7	-0.0262480
PC8	-0.2160035
PC9	0.2330335
PC10	0.0262173
PC11	0.1841956
PC12	-0.3486775
PC13	-0.1612616
PC14	1.9344655
PC15	-0.9977600
PC16	-0.2781283
PC17	-0.2119508
PC18	-0.2338093
PC19	0.0498109
PC20	0.1788627
PC21	-0.0491673
PC22	-0.0881785
PC23	-0.1660471
PC24	0.2349879
PC25	0.4165255

Appendix 4. Intercept and variable coefficients for final model fit onto training data

Code

```
# library(plyr)
#
# data <- join(TRAIN_trad, TRAIN_adv, by = intersect(names(TRAIN_trad[,1:4]), names(TRAIN_adv[,1:4])), 
# data <- join(data, TRAIN_usg, by = intersect(names(data[,1:4]), names(TRAIN_usg[,1:4])), match = 'fir
#
# library(WriteXLS)
# setwd('~/Desktop')
# WriteXLS(data, 'train.xlsx')
#
# final.data <- merge(train_copy, bball_ref_data, by.x = c("PLAYER","SEASON"),
#                     by.y = c("Player","Season"), sort = FALSE)
#
# WriteXLS(final.data, 'train_master.xlsx')


library(readxl)
train <- read_excel("train_master copy.xlsx")

train[,7:8] <- lapply(train[,7:8], factor)

my.data <- train
my.data[,9:56] <- scale(my.data[,9:56])

fit <- glm(`W/L` ~.-Pos, data = my.data[,7:56], family = "binomial")

prin_comp <- prcomp(my.data[,9:56], scale. = F)

# biplot(prin_comp, scale = 0)
library(ggfortify)
autoplot(prin_comp, data = my.data, colour = 'Pos', loadings=TRUE, loadings.colour='black',
         loadings.label = TRUE, loadings.label.colour = 'blue', loadings.label.size = 4)

loadings = prin_comp$rotation

pr_var <- (prin_comp$sdev)^2
prop_varex <- pr_var/sum(pr_var)
par(mgp=c(2.5,1,0))
plot(cumsum(prop_varex), xlab = "Principal Component",
       ylab = "Explained Variance",
       type = "l",lwd=2.2,col='cadetblue')

pca.data <- data.frame(`W/L` = my.data$`W/L`, Pos = my.data$Pos, prin_comp$x[,1:46])

x=model.matrix(W.L~.,pca.data)[,-1]
y=pca.data$W.L

# pca.fit <- glm(W.L ~., data = pca.data, family = "binomial")

# library(MASS)
#
# par(mgp=c(2.5,1,0))
```

```r
# boxplot(`%3PA`~Pos, data = train, xlab='Position', ylab='%3PA', cex.axis = 1.1, cex.lab=1.25)

library(glmnet)
set.seed(19)
cvtfit = cv.glmnet(x, y, family="binomial", nfolds=10, alpha=1, intercept=TRUE, standardize=FALSE)
plot(cvtfit)

bhatL <- coef(cvtfit$glmnet.fit,s=cvtfit$lambda.min)[,1]

response <- predict(cvtfit, x,
                    s = 'lambda.min',
                    type = "response")


nTrain = floor(nrow(my.data)*0.75)
rand = runif(20)
n_components = 10:35
params = matrix(0, 26, 10)
AUCs = matrix(0, 26, 10)
library(pROC)

for (i in 1:length(n_components)) {

  pca.data <- data.frame(`W/L` = my.data$`W/L`, Pos = my.data$Pos, prin_comp$x[,1:n_components[i]])

  x=model.matrix(W.L~.,pca.data)[,-1]
  y=pca.data$W.L
  set.seed(19)
  glm.fit = cv.glmnet(x, y, family="binomial", nfolds = 10, alpha = 1,
                      intercept=TRUE, standardize=FALSE)
  lambda = params[i,] = seq(from=glm.fit$lambda.min,
                            to=glm.fit$lambda.1se,
                            length.out = 10)

  for (l in 1:length(lambda)) {
    test.auc <- rep(0, times=20)
    for (j in 1:20) {
      set.seed(rand[j])
      ii = sample(1:nrow(my.data),nTrain)
      xtrain = x[ii,]; ytrain = y[ii]
      xtest = x[-ii,]; ytest = y[-ii]
      test.fit <- glmnet(xtrain, ytrain, alpha = 1, intercept = TRUE,
                         standardize = FALSE, family = "binomial")
      phat <- predict(test.fit, xtest, s = lambda[l], type = "response")
      roc_obj = pROC::roc(ytest, phat)
      test.auc[j] = pROC::auc(roc_obj)

      # CV.fit <- glmnet(x[folds!=j,], y[folds!=j], alpha = 1,
      #                  intercept = TRUE, standardize = FALSE, family = "binomial")
      # phat <- predict(CV.fit, x[folds==j,],
      #                    s = lambda[l], type = "response")
      # roc_obj = pROC::roc(y[folds==j], phat)
      # CV.auc[j] = pROC::auc(roc_obj)
```

```
    }

  AUCs[i,l] = mean(test.auc)

  }
}

aveAUC = rowMeans(AUCs)

nTrain = floor(nrow(my.data)*0.75)
rand = runif(20)
n_components2 = 35:45
params2 = matrix(0, 11, 10)
AUCs2 = matrix(0, 11, 10)
library(pROC)

for (i in 1:length(n_components2)) {

  pca.data <- data.frame(`W/L` = my.data$`W/L`, Pos = my.data$Pos, prin_comp$x[,1:n_components2[i]])

  x=model.matrix(W.L~.,pca.data)[,-1]
  y=pca.data$W.L
  set.seed(19)
  glm.fit = cv.glmnet(x, y, family="binomial", nfolds = 10, alpha = 1,
                      intercept=TRUE, standardize=FALSE)
  lambda = params2[i,] = seq(from=glm.fit$lambda.min,
                             to=glm.fit$lambda.1se,
                             length.out = 10)

  for (l in 1:length(lambda)) {
    test.auc2 <- rep(0, times=20)
    for (j in 1:20) {
      set.seed(rand[j])
      ii = sample(1:nrow(my.data),nTrain)
      xtrain = x[ii,]; ytrain = y[ii]
      xtest = x[-ii,]; ytest = y[-ii]
      test.fit <- glmnet(xtrain, ytrain, alpha = 1, intercept = TRUE,
                         standardize = FALSE, family = "binomial")
      phat <- predict(test.fit, xtest, s = lambda[l], type = "response")
      roc_obj = pROC::roc(ytest, phat)
      test.auc2[j] = pROC::auc(roc_obj)

      # CV.fit <- glmnet(x[folds!=j,], y[folds!=j], alpha = 1,
      #                  intercept = TRUE, standardize = FALSE, family = "binomial")
      # phat <- predict(CV.fit, x[folds==j,],
      #                    s = lambda[l], type = "response")
      # roc_obj = pROC::roc(y[folds==j], phat)
      # CV.auc[j] = pROC::auc(roc_obj)
    }

    AUCs2[i,l] = mean(test.auc2)

  }
```

```
}

aveAUC2 = rowMeans(AUCs2)


n_components_train = 10:45
params_train = matrix(0, 36, 10)
AUCs_train = matrix(0, 36, 10)
library(pROC)

for (i in 1:length(n_components_train)) {

  pca.data <- data.frame(`W/L` = my.data$`W/L`, Pos = my.data$Pos, prin_comp$x[,1:n_components_train[i]]

  x=model.matrix(W.L~.,pca.data)[,-1]
  y=pca.data$W.L
  set.seed(19)
  glm.fit = cv.glmnet(x, y, family="binomial", nfolds = 10, alpha = 1,
                      intercept=TRUE, standardize=FALSE)
  lambda = params_train[i,] = seq(from=glm.fit$lambda.min,
                              to=glm.fit$lambda.1se,
                              length.out = 10)

  for (l in 1:length(lambda)) {
      phat <- predict(glm.fit, x, s = lambda[l], type = "response")
      roc_obj = pROC::roc(y, phat)
      AUCs_train[i,l] = pROC::auc(roc_obj)
  }
}

aveAUC_train = rowMeans(AUCs_train)

plot(15:45,c(aveAUC[-c(1:5,26)],aveAUC2),type = 'b',col='blue', ylim = c(.88,.90), lwd=2,
     xlab='Principal Components', ylab = 'AUC')
lines(15:45, aveAUC_train[-c(1:5)], type = 'b', col='black', lwd=2)
abline(v=25, col='red',lwd=2)
legend("bottomright",legend = c('Training','Testing'),col=c('black','blue'),pch = 1, lwd = 2)

# Confirmation

pca_25 = data.frame(`W/L` = my.data$`W/L`, Pos = my.data$Pos, prin_comp$x[,1:25])

x_25=model.matrix(W.L~.,pca_25)[,-1]
y_25=pca_25$W.L
fit_25 = cv.glmnet(x_25, y_25, family="binomial",
                   nfolds=10, alpha=1, type.measure = 'auc',
                   intercept=TRUE, standardize=FALSE)
plot(fit_25)

cat(max(fit_25$cvm),'\n',fit_25$lambda.min)

final_coefs <- as.data.frame(coef(fit_25$glmnet.fit,s=fit_25$lambda.min)[,1])
names(final_coefs)=""
```

```r
final_coefs <- as.data.frame(t(final_coefs))


# fit_25 (principal components = 25) is optimal model




TEST_trad <- read_excel("projectdata.xlsx",
                        sheet = "TEST_trad")
TEST_adv <- read_excel("projectdata.xlsx",
                        sheet = "TEST_adv")
TEST_usg <- read_excel("projectdata.xlsx",
                        sheet = "TEST_usg")
bball_ref_test <- read_excel("bball_ref_test.xlsx")

library(plyr)
merge <- join(TEST_trad, TEST_adv, by = intersect(names(TEST_trad[,1:4]), names(TEST_adv[,1:4])), match
merge <- join(merge, TEST_usg, by = intersect(names(merge[,1:4]), names(TEST_usg[,1:4])), match = 'firs

final.test <- merge(merge, bball_ref_test, by.x = c("PLAYER"),
                    by.y = c("Player"), sort = FALSE)

library(WriteXLS)
setwd('~/Desktop')
WriteXLS(final.test, 'test_master.xlsx')

test <- read_excel("test_master.xlsx")
test[,5:6] <- lapply(test[,5:6], factor)

test_scale <- test
test_scale[,7:54] <- scale(test[,7:54])

test.data <- predict(prin_comp, newdata = test_scale)

test_25 = data.frame(`W/L` = test$`W/L`, Pos = test$Pos, test.data[,1:25])

x_25=model.matrix(W.L~.,test_25)[,-1]
y_25=test_25$W.L

phat.test <- predict(fit_25, x_25, s = "lambda.min", type = "response")
roc_test = pROC::roc(y_25, phat.test)

plot(1-roc_test$specificities, roc_test$sensitivities,
     type = 'b', col='red', xlab = '1-Specificity', ylab = 'Sensitivity',
     xlim = c(0,1), ylim = c(0,1))
legend(x=.7,y=.175,legend = 'AUC=0.899')

yhat.test = predict(fit_25, x_25, s = "lambda.min", type = "class")

tab = table(y_25, yhat.test)
tpr = tab[4]/(tab[4]+tab[2])
fpr = tab[3]/(tab[1]+tab[3])
```

```
abline(v=fpr,col='blue',lwd=1.5)
abline(h=tpr,col='blue',lwd=1.5)
abline(a=0,b=1,lty=2,lwd=2)

image(tab, col=blues9, xaxt='n',yaxt='n')
box(lwd=5)
text(0,0,"0.098",col = 'white',cex = 2)
text(0,1,"0.400",col = 'black',cex = 2)
text(1,0,"0.401",col = 'black',cex = 2)
text(1,1,"0.101",col = 'white',cex = 2)
mtext('Observed', side=3, line=1.5, at=.5)
mtext('Predicted', side=2, line = 1.5, at=.5)
mtext('L', side=3, line=.5, at=0, font = 2, cex = 2, col = 'red')
mtext('W', side=3, line=.5, at=1, font = 2, cex = 2, col = 'forestgreen')
corners = par("usr") #Gets the four corners of plot area (x1, x2, y1, y2)
par(xpd = TRUE) #Draw outside plot area
text(x = -.60, y = 1, "L", font = 2, col = 'red',cex = 2)
text(x = -.60, y = 0, "W", font = 2, col = 'forestgreen',cex = 2)

allAUC = data.frame(allAUC, row.names = as.character(10:45))
all.params = rbind(params[-26,],params2)
all.params = data.frame(all.params, row.names = as.character(10:45))

new_coefs = matrix(0,48,25)
for (i in 1:25){
  new_coefs[,i] = loadings[,i] * as.numeric(final_coefs[i+5])
}

simp_coefs <- as.data.frame(rowSums(new_coefs), row.names = names(test[,7:54]))
var_imp <- abs(simp_coefs)
```